NS3 Project Update 1

Algorithm Implementation Overview

SAIF AHMED KHAN
ID: 1705110

CSE 322

Reference Paper

Updated Congestion Control Algorithm for TCP Throughput improvement in Wired and Wireless Network

Global Journal of Computer Science and Technology
Vol. 9 Issue 5 (Ver 2.0), January 2010.

Authors:
Prof.K.Srinivas
Dept of MCA,KIPGS,KONDAIR,509125 India; srinivas_mnb4u@yahoo.co.in
Dr.A.A.Chari
Dept of SQC,Rayalaseema University, Kurnool 518002, India; directorresearch.ru@gmail.com,
Prof.N.Kasiviswanath
Dept of CSE, GPREC, Kurnool 518002, India; nkv@gmail.com

Creating New Models

In path “ns-3.35/src/internet/model” create two new files named “tcp-constant.cc” and “tcp-constant.h”

Add the file name “tcp-constant.cc” in the list **obj.source**

wscript

ns-3.35 > src > internet > wscript

```
105 def build(bld):
106     # bridge dependency is due to global routing
107     obj = bld.create_ns3_module('internet', ['bridge', 'traffic-control', 'network', 'core'])
108     obj.source = [
109         'model/ip-l4-protocol.cc',
110         'model/udp-header.cc',
111         'model/tcp-header.cc',
112         'model/ipv4-interface.cc',
113         'model/ipv4-l3-protocol.cc',
114         'model/ipv4-end-point.cc',
115         'model/udp-l4-protocol.cc',
116         'model/tcp-l4-protocol.cc',
```

Creating New Models

Add the file name “tcp-constant.h” in the list **headers.source**

wscript

ns-3.35 > src > internet > wscript

```
323     headers = bld(features='ns3header')
324     headers.module = 'internet'
325     headers.source = [
326         'model/udp-header.h',
327         'model/tcp-header.h',
328         'model/tcp-option.h',
329         'model/tcp-option-winscale.h',
330         'model/tcp-option-ts.h',
331         'model/tcp-option-sack-permitted.h',
332         'model/tcp-option-sack.h',
333         'model/tcp-option-rfc793.h',
```

The Main Changes: IncreaseWindow

tcp-linux-reno.cc

ns-3.35 > src > internet > model > tcp-linux-reno.cc > {} ns3 > IncreaseWindow(Ptr<TcpSocketState>, u

```
void
TcpLinuxReno::IncreaseWindow (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked
{
  NS_LOG_FUNCTION (this << tcb << segmentsAcked);

  // Linux tcp_in_slow_start() condition
  if (tcb->m_cWnd < tcb->m_ssThresh)
    {
      NS_LOG_DEBUG ("In slow start, m_cWnd " << tcb->m_cWnd << " m_ssThresh "
      segmentsAcked = SlowStart (tcb, segmentsAcked);
    }
  else
    {
      NS_LOG_DEBUG ("In cong. avoidance, m_cWnd " << tcb->m_cWnd << " m_ssThr
      CongestionAvoidance (tcb, segmentsAcked);
    }
}
```

```
if(slow_start_state)
    slow_start(); /* open cwnd by one segment on each ACK arrival */
else
{
    /*fractional increase greater than threshold */
    if(abs(rtt_arc-rtt_var)/rttarc > ß)
    {
        /* recalculate window and archive the value of rtt_var */
        cwnd_ = (Estimated_Bandwidth*rttmin)/ seg_size_
        if(cwnd_ < 1) cwnd_ = 1;
        rttarc = rttvar ;
    }
}
```

The Main Changes: CongestionAvoidance

tcp-linux-reno.cc

ns-3.35 > src > internet > model > tcp-linux-reno.cc > {} ns3 > CongestionAvoidance(Ptr<TcpSocketState>, uint32_t)

```
74  void
75  TcpLinuxReno::CongestionAvoidance (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked)
76  {
77    NS_LOG_FUNCTION (this << tcb << segmentsAcked);
78
79    uint32_t w = tcb->m_cWnd / tcb->m_segmentSize;
80
81    // Floor w to 1 if w == 0
82    if (w == 0)
83      {
84        w = 1;
85      }
86
87    NS_LOG_DEBUG ("w in segments " << w << " m_cWndCnt " << m_cWndCnt << " segments acked " << segmentsAcked);
88    if (m_cWndCnt >= w)
89      {
90        m_cWndCnt = 0;
91        tcb->m_cWnd += tcb->m_segmentSize;
92        NS_LOG_DEBUG ("Adding 1 segment to m_cWnd");
93      }
```

Testing TCPConstant

To set the default socket type before any internet stack-related objects are created, one may put the following statement at the top of the simulation program:

```
Config::SetDefault ("ns3::TcpL4Protocol::SocketType", StringValue ("ns3::TcpNewReno"));
```

seventh.cc

ns-3.35 > scratch > seventh.cc > main(int, char * [])

```
int
main (int argc, char *argv[])
{
  //changing default Congestion Control Algo by adding this line:
  Config::SetDefault ("ns3::TcpL4Protocol::SocketType", StringValue ("ns3::TcpConstant"));

  bool useV6 = false;
```

Run the file and then uses **GNUPlotter** to see results of the .cwnd file generated on a graph

Thank You